456-TP-007-001

ECS Project Training Material Volume 7: Database Administration

Technical Paper

May 1997

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Mike Resnick Date
EOSDIS Core System Project

SUBMITTED BY

Tom Hickey M&O Deputy Manager Date
EOSDIS Core System Project

Hughes Information Technology Systems
Upper Marlboro, Maryland

This page intentionally left blank.

Abstract

This is Volume 7 of a series of 10 volumes containing the training material for the Pre-Release B Testbed of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). This lesson provides a detailed description of the process required to perform the tasks associated with database administration.

***Keywords**:* training, database administration, course objective, metadata

This page intentionally left blank.

Contents

Database Segments

Transaction Logs

Backup and Recovery

SQL Server Login Accounts and Privileges

Database Tuning and Performance Monitoring

Integrity Monitoring

Troubleshooting

Practical Exercises

Slide Presentation

Figure

Table

Introduction

Identification

Training Material Volume 7 is part of a series of Technical Papers that will be used to teach Maintenance and Operations (M&O) concepts to the M&O staff at the following Distributed Active Archive Centers (DAACs): Langley Research Center (LaRC), National Snow and Ice Data Center (NSIDC) and EROS Data Center (EDC).

Scope

Training Material Volume 7: Database Administration is designed to provide the operations staff with sufficient knowledge and information to satisfy all lesson objectives.

This document reflects the August 23, 1995 Technical Baseline maintained by the contractor Configuration Control Board (CCB) in accordance with ECS technical direction #11, dated December 6, 1994.

Purpose

The purpose of this Technical Paper is to provide a detailed course of instruction that forms the basis for understanding database administration. Lesson objectives are developed and will be used to guide the flow of instruction for this lesson. The lesson objectives will serve as the basis for verifying that all lesson topics are contained within this Student Guide and slide presentation material.

Organization

This document is organized as follows:

Introduction:	The Introduction presents the document identification, scope, purpose, and organization.
Student Guide:	The Student Guide identifies the core elements of this lesson. All Lesson Objectives and associated topics is included.
Slide Presentation:	Slide Presentation is reserved for all slides used by the instructor during the presentation of this lesson.

This page intentionally left blank.

Database Administration

Lesson Overview

This lesson will provide you with the tools needed to perform the various tasks required to administer and maintain the database and structure management for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) during maintenance and operations.

Lesson Objectives

Overall Objective - This lesson provides a detailed description of the different tasks required to maintain the database and structure management for ECS, provide the operations interface to perform database administration utilities such as product installation and disk storage management, managing user accounts and privileges, backup and recovery, monitoring physical allocation of database resource information, loading metadata and maintaining metadata.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures required to perform database administration.

Specific Objective 1 - The student will be able to create new database devices, allocate appropriate disk space to house the new database, and maintain database segments including:

Indexing physical allocation of databases.

Managing and monitoring the use of available disk space, memory, connection error logs, state of transaction logs, device problems, etc.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of new database devices, the allocation of appropriate disk space, and maintenance of database segments.

Specific Objective 2 - The student will able to start and shutdown the SQL server.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating the startup and shutdown of the SQL server.

Specific Objective 3 - The student will be able to perform database user account and access privilege procedures including:

- Creating user accounts.
- Granting and revoking access privileges for data retrieval, insertion, deletion and update of objects.
- Granting and revoking roles for SQL server users groups.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of user accounts and the granting and revoking of access privileges.

Specific Objective 4 - The student will be able to perform database security and auditing procedures.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database security and auditing procedures.

Specific Objective 5 - The student will be able to perform database integrity monitoring.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database integrity monitoring.

Specific Objective 6 - The student will be able to perform full and incremental database backups on a regular or on-demand basis.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to full and incremental database backups.

Specific Objective 7 - The student will be able to perform a recovery of the database following a system failure or on-demand.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database recovery.

Specific Objective 8 - The student will be able to configure databases unique to ECS DAACs including:

- Making database size estimates and planning.
- Preparing database-unique attributes.
- Preparing database reports.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database configuration specific to the ECS.

Specific Objective 9 - The student will be able to perform database tuning and performance monitoring procedures including:

- Design and indexing.
- Responding to queries.
- Monitoring and boosting performance.

Condition - The student will be given a copy of *456-TP-007-001 ECS Project Training Material Volume 7: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database tuning and performance monitoring.

Importance

ECS relies on vast amounts of data from the science user's perspective and from a maintenance and operations perspective. Accurate information stored in the science databases allows science user's to access necessary data quickly. Similarly, databases that maintain operational data must be kept current in order for routine and specialized administrative tasks to be performed.

Special Instructions on Procedures

All procedures in this training guide assume that you are logged in to a system server or workstation as yourself.

Many DBA tasks will be performed by using scripts that are run from the command line. Some scripts may require modification which will require you to use a text editor of your choice. These procedures do not give keystroke-by-keystroke information on how to edit the file; they only instruct you to make the changes required to perform the task.

This page intentionally left blank.

Overview

Database administration involves the maintenance of databases by installing SQL server products, managing disk storage space, managing user accounts and privileges, and performing database backup and recovery operations.

DBA Tasks and Functions

The Database Administrator (DBA), is the individual or office responsible for the installation, configuration, update, maintenance, and overall performance and reliability of the SQL Server database. In general, the DBA is concerned with the availability of the server, the definition and management of resources allocated to the server, the definition and management of databases and objects resident on the server, and the relationship between the server and the operating system.

The DAAC Database Administrators perform the following functions:

- Perform the database administration utilities, such as database backup, maintenance of database transaction logs, and database recovery.
- Monitor and tune the physical allocation of database resources.
- Maintain user accounts.
- Create user registration and account access control permissions in the security database.
- Work with data specialists in information management tasks involving databases, data sets, and metadata management.
- Perform daily database synchronization.

Databases To Be Administered

Databases that the DBA will administer during Testbed operations include but are not limited to:

- Autosys databases (*autosys_srvr* and *autosys_backup*)
- Planning and Data Processing Segment databases (*pdps_db_ops_srvr*, *pdps_db_ops_backup*)
- Sybase internal databases (*master*, *model*, *sybsystemprocs*, *sybsecurity*)
- Other special databases to track various processes at the direction and discretion of the System Administrator.

Sybase Directory Structure

The following table describes the directory structure used for Sybase database administration in the Pre-Release B Testbed.

$SYBASE is a shortcut for the full path to the Sybase home directory. Because each DAAC may support a different full path to that directory (e.g., /usr/ecs/Rel_A/COTS/sybase, /vol0/testbed/COTS/sybase, etc.), this Technical Paper will refer to the path as **$SYBASE**.

Table 1. Sybase Directory Structure.

Directory	Contains
$SYBASE/bin	Utilities necessary to load, run, and access the server
$SYBASE /install	Files used to start dataservers, backupserver and to record server messages (errorlogs)
$SYBASE /lib	db-lib, ct-lib, and xa-lib client library files used by applications to gain access to the server
$SYBASE /scripts	Root directory for all script files executed on the server
$SYBASE /scripts/create.databases	SQL script files used to create databases on the server
$SYBASE /scripts/create.db_objects	SQL script files used to create tables, indexes, triggers, rules, etc., in user databases
$SYBASE /scripts/create.devices	SQL script files (disk init) used to map physical storage to a logical name
$SYBASE /scripts/create.procedures	T-SQL file used to define stored procedure objects in user databases
$SYBASE /scripts/create.users	SQL command files used to define logins and add them to user databases
backup directory	Root directory that contains all backup subdirectories. *It is recommended, but not required, that this directory be on a separate physical disk*
backup subdirectories	Created each evening by the RUN_backups cron job, they are named backups_for_YYMMDD and contain the following files: servername_dbname.dat - full database dumps servername_dbname_tx.dat - transaction file dumps (where applicable) servername_backup_log.YYMMDD servername_srvr_stmts.out servername_srvr_stmts.sql backup_summary.YYMMDD

Naming Conventions

In order to keep track of the large number of files that you will create and work with throughout your tenure as DBA, and because other people will come into contact with these files at one time or another, please follow these simple naming conventions for your files:

- The file name should indicate the function and/or content of the object regardless of the length of the file name.
- Only easily understandable abbreviations should be used.
- Parts of names are separated by the underscore character (_).
- Only one optional suffix is permitted and is appended to the file name by a period (.).
- The full path of the object is considered to be part of the name.

Some examples of these files are as follows:

The testbed has two SQL Servers installed: one for the Planning & Data Processing System (PDPS) application, and one for Autosys. The servers themselves are named for the applications they support (e.g., **autosys1_srvr**) or machines they are on (e.g., **plng2sun_srvr**).

All backup files are located under the $SYBASE/backups directory, which may be on a separate physical disk. These directories are kept for a period of 30 days and they are named **backups_for_YYMMDD** where **YYMMDD** is the sortable six-digit year, month, and day. For example, a backup created on April 23, 1997 would be in a backup directory called **backups_for_970423**.

All SQL script files have the **.sql** suffix. Their names reference the objects they create or functions they perform, and are all located either in $SYBASE/scripts or below. A SQL file to create the pdps_db_ops database is named **$SYBASE/scripts/create.databases/pdps_db_ops.sql**, and another file used to alter the database might be called **$SYBASE/scripts/create.databases/alter_pdps_db_ops.sql**.

This page intentionally left blank.

Starting and Stopping SQL Server

Start an SQL Server Process

A SQL server process is started when a new server is installed or after a system outage.

In order to perform the procedure, the DBA must have obtained the database server name.

To start a SQL server process, the DBA must have sa_role (SQL Server)

Start the SQL Server Process Procedure

1 Log into the server on which the new user database is to be created by typing: **telnet** ***ServerName*** or **rlogin** ***ServerName*** or **rsh** ***ServerName***, then press **Return**.

- If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
- If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY** ***IPNumber*****:0.0** or **setenv DISPLAY** ***ServerName*****:0.0**, then press **Return**.

5 At the UNIX prompt, type **startserver -f** ***runserver_file*** **-c** ***configuration_file***, then press **Return**.

6 At the UNIX prompt, type **showserver**, then press **Return**.

- This displays a listing of the SQL Server processes that are running.

Stop an SQL Server

A SQL server process is stopped by the DBA when the system to be brought down for maintenance. The **shutdown** command issued from within ISQL shuts down the SQL server on which you are logged in. It allow for the completion of any current SQL processes and blocks the start of any new SQL processes before the server is shutdown. Adding a **nowait** option to the command immediately terminates all processes and shuts down SQL server. Using the **nowait** option may result in loss of data and a more complicated recovery procedure, so use it sparingly.

Stop the SQL Server Procedure

1 Log into the server on which the new user database is to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.

- If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
- If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

5 At the UNIX prompt, type **su sybase**, then press **Return**.

- A password prompt is displayed.

6 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/Rel_A/COTS/sybase** and all required environment variables have been set.

7 At the UNIX prompt, type **shutdown**, then press **Return**.

8 At the UNIX prompt, type **showserver**, then press **Return**.

- This displays a listing of the SQL Server processes that are running.
- There should be no SQL Server processes running.

Database Devices

A database device stores the objects that make up databases. The term *device* does not necessarily refer to a distinct physical device; it can refer to any piece of disk, such as a partition; or a file in the file system used to store databases and their objects (Figure 1).

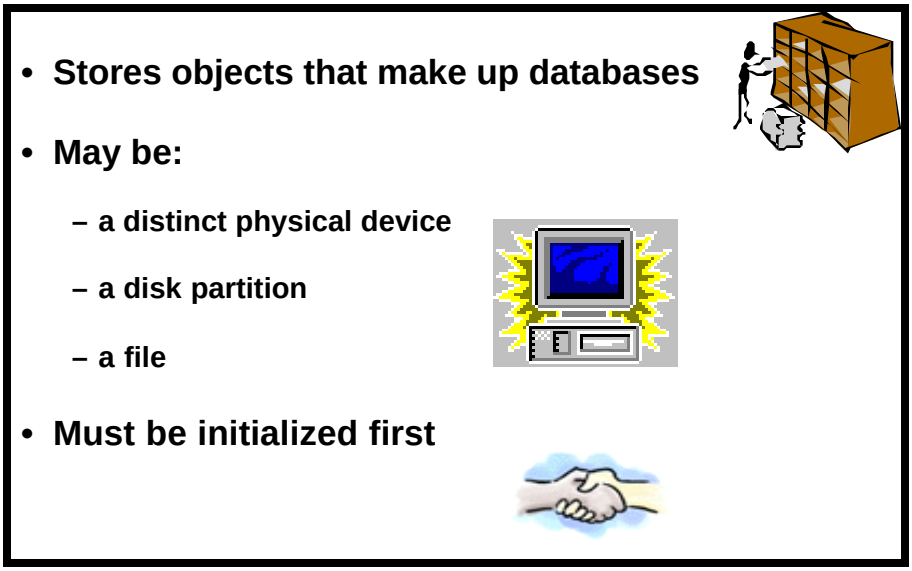

Figure 1. Database Devices

Each database device or file must be prepared and made known to SQL Server before it can be used for database storage, a process is called initialization. Once a database device is initialized, it can be:

- Allocated to the pool of space available to a user database.
- Allocated to a user database, and assigned to store a specific database object or objects.
- Used to store a database's transaction logs.
- Designated as a default device for **create** and **alter** database commands.

A database device is created when the System Administrator determines that new disk space is available for use by a Sybase database, or as part of the Database Recovery Procedure. The System Administrator makes a request to the DBA who creates the new database device and notifies the System Administrator when the device has been created.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database device to create.

- Physical device on which to place database device.
- Device size in megabytes.
- For a mirrored device, name of the mirror device.

Create Database Device Procedure

---

1 Log into the server on which the new database devices it to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.

- If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
- If you use **rlogin** or **rsh**, the system uses the same logname currently in use and prompts you for a password. Skip to Step 3 below.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

For each database device to be created, perform Steps 5 through yyy:

5 Log into Sybase by typing: **su sybase**, then press **Return.**

- A password prompt is displayed.

6 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/Rel_A/COTS/sybase** and all required environment variables have been set.

7 At the UNIX prompt, type **cd scripts/create.devices**, then press **Return.**

- This places you in the directory that contains all the scripts used to create database devices.

- It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see *Restore Database Devices*).

8 At the UNIX prompt, type **cp template.sql *DeviceName*.sql**, then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the database device.

9 Using the text editor of your choice, edit ***DeviceName*.sql** (Figure 2) and make changes to information enclosed in brackets (including the brackets) as appropriate:

```
/*******************************************************/
/* name: [template.sql]                                */
/* purpose:                                            */
/* written:                                            */
/* revised:                                            */
/* reason:                                             */
/*******************************************************/
disk init name = [device name],
physname = "/dev/[device name]",
vdevno = [#],
size = [size]
```

Figure 2. Sample emplate.sql File for Creation of a Database Device

- In the area delimited by **/* */**, enter an appropriate description of the script including the file name, date written and person who wrote the script, it's purpose, and any other information deemed appropriate. Be sure to enclose each line of the comment between **/* */**.

- The **disk init name** is the ***DeviceName*** is used in Step 8 above.

  – You may not use spaces or punctuation except the underscore character (_) in ***DeviceName***.

  – Remember that the name you assign is case sensitive.

  – Be sure there is a comma after ***DeviceName***

- The **physname** is the ***FullPath_to_DeviceName***..

  – Be sure to enclose ***FullPath_to_DeviceName*** in double quotes.

  – Be sure to place a comma after ***FullPath_to_DeviceName*** but outside the double quotes.

- The size is the ***DeviceSize*** in bocks.

- To compute the number of blocks, multiply the device size in megabytes by 512; e.g., a 1,000 Mb device has 512,000 blocks
- Be sure to place a comma after ***DeviceSize***.

- The **vdevno** is the ***VirtualDeviceNumber***
  - **vdevno** is a unique identifying number for the database device. It must be unique among the all devices used by SQL Server and is never reused. Device number 0 represents the device named **d_master** that stores the system catalogs. Legal numbers are between 1 and 255, but the highest number must be one less than the number of database devices for which your system is configured. For example, for a system with the default configuration of 10 devices, the legal device numbers are 1-9. The next available number can be determined by looking at the output from the **sp_helpdevice** command and selecting the next number in sequence. If you use an existing number, Sybase will return the message, "device activation error."

10 After the changes have been made, save the file according to the rules of your text editor.

11 At the UNIX prompt, type:

isql -Usa -S*ServerName* -i*DeviceName*.sql -o*DeviceName*.out

then press **Return**.

- ***ServerName*** is the name of the server on which the database device will be created.
- ***DeviceName.sql*** is the name of the script file you created in step 8.
- ***DeviceName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

12 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

13 When the UNIX prompt is again displayed the process is complete.

14 At the UNIX prompt, type **more *DeviceName.out*** , then press **Return**.

- This allows you to view the ***DeviceName.out*** file to confirm that the device has been created or to check for device creation errors.

A sample of a completed Database Device creation script follows Figure 3).

```
/*****************************************/
/* name: datadev1.sql                    */
/* purpose: tracking training program    */
/* written: 4/25/97 mresnick             */
/* revised:                              */
/* reason:                               */
/*****************************************/
disk init name = datadev1,
physname = “/dev/rdsk/c0t0d4s3“,
vdevno = 5,
size = 5120
```

Figure 3. Completed Database Cevice Creation Script

This page intentionally left blank.

User Databases

A user database is created when an approved request is received by the DBA, or as part of the Database Recovery Procedure (see **Database Recovery)**. The database name, approximate size of the database and the transaction log are necessary information for the DBA to complete the task of creating a user database. It is the user's responsibility to determine the appropriate database design.

The requester fills out a "User Database Request Form" and submits it to the supervisor.

The requester's supervisor reviews the request, and if s/he determines that it is appropriate to create the User Database, forwards the request to the Operations Supervisor (Ops Super). The Ops Super verifies that all required information is contained on the form. (Incomplete forms are returned to the requester's supervisor for additional information.) If it is complete and if the request for a new User Database fits within policy guidelines, the Ops Super approves the request and forwards the request form to the DBA to implement. After the User Database is created, the DBA notifies the requester that the new database is available. The DBA also sends an e-mail message to the user's supervisor informing them that the new User Database was created.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database to create.
- Size of database and the transaction log.
- Database devices to use.
- To create a database, the DBA must have sa_role (SQL Server) privileges.

Create User Database Procedure

1 Log into the server on which the new user database is to be created by typing: **telnet** ***ServerName*** or **rlogin** ***ServerName*** or **rsh** ***ServerName***, then press **Return**.

- If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
- If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.

- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

5 Log into Sybase by typing: **su sybase**, then press **Return.**

- A password prompt is displayed.

6 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/Rel_A/COTS/sybase** and all required environment variables have been set.

7 At the UNIX prompt, type **cd scripts/create.databases**, then press **Return.**

- This places you in the directory that contains all the scripts used to create database devices.
- It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see **Restore Databases**).

8 At the UNIX prompt, type **cp template.sql *DBName*.sql**, then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the user database.

9 Using the text editor of your choice, edit ***DBName*.sql** (Figure 4) and make changes to information enclosed in brackets (including the brackets) as appropriate:

- **DBName** is the name of the database and should describe it's function succinctly.
- **DBDeviceName** is the name of the existing, approved database device on which **DBName** will reside.
- **DBSize_in_MB** is the estimated size of the database in megabytes.
- **LogDBDeviceName** is the name of the database device holding the transaction log
- **LogSize_in_MB** is the estimated size of the transaction log in megabytes.
- The **sp_helpdb** command provides feedback at the end of the script to assure that the database has been created.

```
/*****************************************/
/* name: [template.sql]                  */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/
create database [DBName]
on [DBDeviceName] = [DBSize_in_MB]
log on [LogDBDeviceName] = [LogSize_in_MB]
go
sp_helpdb [DBName]
go
```

Figure 4. Sample template.sql File for Creation of a Database

10 After the changes have been made, save the file according to the rules of the text editor.

11 At the UNIX prompt, type:

isql -Usa -S*ServerName* -i*DBName*.sql -o*DBName*.out

then press **Return**.

- ***ServerName*** is the name of the server on which the database device will be created.
- ***DBName.sql*** is the name of the script file you created in step 9.
- ***DBName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

12 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

- Remember the ***SybaseSAPassword*** is case sensitive.

13 When the UNIX prompt is again displayed the process is complete.

14 At the UNIX prompt, type **more *DBName.out*** , then press **Return**.

- This allows you to view the ***DBName.out*** file to confirm that the device has been created or to check for database creation errors.

A sample of a completed Create Database script follows (Figure 5).

```
/*****************************************/
/* name: mydb.sql                        */
/* purpose: track specific data items    */
/* written: 4/26/97 mresnick             */
/* revised:                              */
/* reason:                               */
/*****************************************/
create database mydb
on datadev1 = 10
log on datadev1 = 2
```

Figure 5. Completed Create Database Script

Database Segments

Creating Segments

A **segment** is a collection of the database devices and/or fragments available to a particular database. Tables and indexes can be assigned to a particular segment, and thereby to a particular physical device, or can span a set of physical devices.

Segments are named subsets of the database devices available to a particular SQL Server database. A segment can best be described as a label that points to one or more database devices. Segment names are used in **create table** and **create index** commands to place tables or indexes on specific database devices. The use of segments can increase SQL Server performance, and can give the System Administrator or Database Owner increased control over placement, size and space usage of specific database objects. For example:

- If a table is placed on one device, and its non-clustered indexes on a device on another disk controller, the time required to read or write to the disk can be reduced, since disk head travel is usually reduced.

- If a large, heavily-used table is split across devices on two separate disk controllers, read/write time may be improved.

SQL Server stores the data for text and image columns on a separate chain of data pages. By default, this text chain is placed on the same segment as the table. Since reading a text column requires a read operation for the text pointer in the base table and an additional read operation on the text page in the separate text chain, placing the text chain on a separate physical device can improve performance.

If you place tables and indexes only on specific segments, those database objects cannot grow beyond the space available to the devices represented by those segments, and other objects cannot contend for space with them.

Segments can be extended to include additional devices as needed.

You can use thresholds to warn you when space becomes low on a particular database segment.

Segments are created within a particular database from the database devices already allocated to that database. Each SQL Server database can contain up to 32 segments. The database devices must first be initialized with **disk init**, and then be made available to the database with a **create database** or **alter database** statement before you can assign segment names.

When you first create a database, SQL Server creates three segments in the database (Table 2):

Table 2. Default segment names and functions.

Segment	Function
system	Stores this database's system tables.
logsegment	Stores this database's transaction log.
default	Stores all other database objects-unless you create additional segments, and store the table or index on the new segments by using **create table... on *segment_name*** or **create index... on *segment_name*.**

Database segments are created when a database is created, when the Database Administrator determines that it is necessary to allocate database objects to a new or additional devices, or as part of the Database Recovery Procedure.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database
- Database device extents
- Space on the database allocated to the Database device

To create a database, the DBA must have sa_role.

Create Database Segment Scenario and Procedure

The DBA receives a request to create a segment for the storage of the DATA_INFO table indexes in the pdps_db_ops database, on a separate physical disk. Two devices **data_dev1** and **data_dev2** have already been created and are located on different physical disks.

1 At the UNIX prompt, type **cd /scripts/create.segments**, then press **Return**.

2 At the UNIX prompt, type **cp template.sql *Segment*.sql**, then press **Return**.

3 Using the text editor of your choice, edit ***Segment*.sql** (Figure 4) and make changes to information enclosed in brackets (including the brackets) as appropriate:

```
/*****************************************/
/* name: [template.sql]                  */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/

sp_addsegment [seg_name], [DBname], [DBDevice]
go
```

Figure 6. Create Database Segment Template File.

4 After the changes have been made, save the file according to the rules of the text editor.

5 At the UNIX prompt, type:

isql -Usa -S*servername* -i*Segment*.sql -o*Segment*.out

then press **Return.**

- The system will prompt you for a password.

6 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

- Remember the ***SybaseSAPassword*** is case sensitive.

7 When the UNIX prompt is again displayed the process is complete.

8 At the UNIX prompt, type **more *Segment.out*** , then press **Return**.

- This allows you to view the ***Segment.out*** file to confirm that the device has been created or to check for database creation errors.

This page intentionally left blank.

Transaction Logs

What is a Transaction?

A **transaction** is a the SQL Server's basic unit of work. Each SQL statement that modifies data in a database is considered a transaction, and SQL Server uses them to keep track of all database changes. A transaction consists of one or more Transact-SQL statements that succeed or fail as a unit. Each success or failure is recorded automatically in the database's transaction log.

What is a Transaction Logs?

The Transaction Log is an ever-expanding file that records each and every transaction that occurs in a database. This log is used to perform a complete recovery of the database in the event of a media failure. The transaction log is normally maintained on a different database device or even a completely separate system than the rest of the database so that it can be read should a failure occur.

How Transaction Logging Works

The transaction log is a write-ahead log. When a user issues a statement that would modify the database, SQL Server automatically opens the transaction log and writes a **transaction start** statement. After all changes for a statement have been recorded, a **transaction end** statement is written to the log, and the log is written to an in-cache copy of the data page. The data page remains in cache until the memory is needed for another database page at which time the **checkpoint** process writes the changes to disk.

If any statement in a transaction fails to complete due to operator error, the operator aborting the procedure, or mechanical failure, SQL Server automatically reverses all changes made by the transaction. SQL Server writes an "end transaction" record to the log at the end of each transaction, recording the status (success or failure) of the transaction.

Each database has its own transaction log that records all changes to its database which may or may not be on the same database device. It is strongly recommended that transaction logs be kept on a device separate from the database as insurance against catastrophic data loss.

The transaction log grows in size over time and may cause database problems when full. Therefore, the transaction log must be dumped periodically.

Getting Information about the Transaction Log

The transaction log contains enough information about each transaction to ensure that it can be recovered. Use the **dump transaction** command to copy the information it contains to tape or disk. Use **sp_spaceused syslogs** to check the size of the log, or **sp_helpsegment logsegment** to check the space available for log growth.

This page intentionally left blank.

Backup and Recovery

Database and transaction log backups and recoveries are probably the most important tasks the Database Administrator must perform. Without regular and complete backups of databases and transaction logs, the potential for enormous amounts of data to be lost is very great.

Frequency and Schedule

Each DAAC is responsible for determining its own backup schedule to meet its individual requirements. Nonetheless, it is strongly suggested that a regular schedule of database backups be established and maintained. Although the databases are included in the daily backups of the entire testbed system, separate database backups should be performed nightly.

Database backups are run by a UNIX **cron** job which executes the **RUN_backups** program located in the **$SYBASE/scripts/backups** directory. No intervention in the automatic backup process is required by the DBA, though periodic checks of the Backup subdirectories are recommended.

Manual backups can be performed at any time by the DBA and are recommended for the following situations:

- Any change to the **master** database, including new logins, devices, and databases.
- Any major change to user databases, such as a large ingest or deletion of data, definition of indexes, etc.,
- Other mission-critical activities as defined by the DAAC operations controller.

Manual Database Backup Procedure

1 Log into the server on which contains the database to be backed up by typing: **telnet** ***ServerName*** or **rlogin** ***ServerName*** or **rsh** ***ServerName***, then press **Return**.

- If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
- If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

For each database device to be created, perform Steps 5 through yyy:

5 Log into Sybase by typing: **su sybase**, then press **Return.**

- A password prompt is displayed.

6 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/Rel_A/COTS/sybase** and all required environment variables have been set.

7 At the UNIX prompt, type **isql -Usa**, then press **Return**.

8 To backup the database, at the ISQL prompt, type **dump database *DBName* to *BackupDirectory/DBName*.dat**, then press **Return**; type **go**, then press **Return**.

9 To backup the transaction log, at the ISQL prompt, type **dump transaction *DBName* to *BackupDirectory/DBName*_tx.dat**, then press **Return**; type **go**, then press **Return**.

Database Recovery

A database recovery is performed when the Database Administrator determines that some database data is corrupt, or when the System Administrator determines that a database device has failed. The Database Administrator makes a request to the System Administrator, who performs the restore and notifies the Database Administrator when the restore is complete.

The symptoms of media failure are as variable as the causes. If only a single block on the disk is bad, your database may appear to function perfectly for some time after the corruption appears; this is why you should run **dbcc** commands frequently. If an entire disk or disk controller is bad, you will not be able to use a database. SQL Server marks it as suspect and displays a warning message. If the disk storing the master database fails, users will not be able to log into the server, and users already logged in will not be able to perform any actions that access the system tables in master.

The complete database device restoration procedure is actually a suite of procedures that must be performed in the prescribed order:

1. The device failure has been verified by the System Administrator and requests restoration from the DBA.

2. If possible, perform a dump the current database and the current database transaction log for each database on the failed device is backed up. If this is not possible due to the damage to the database device, then the DBA will have to use the most recent database and transaction log dumps.
3. If possible, the DBA examines the space usage for each database on the failed device. The same defaults should be set when the new database device(s) is(are) created.
4. The DBA drops the database(s) on the failed device, then the database device itself is dropped.
5. The DBA initializes a new database device. This is why it is important to keep the device creation scripts.
6. The DBA recreates each user database on the new database device. This is why it is important to keep the database creation scripts.
7. Each database is reloaded with data from the database backups and the transaction log backups. It is this procedure that is detailed below.
8. The DBA notifies the System Administrator when the database restoration is complete.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database device which has failed
- Name of replacement device
- Name of the databases which reside on the failed device
- Name of the backup volumes
- Name of the dump files on the backup volumes

Load Database and Load Transaction commands

Manual recovery of a user database is performed by the System Administrator by the use of the **LOAD DATABASE** and **LOAD TRANSACTION** commands. For issues concerning the **master** database, please consult your System Administrator's Guide for assistance. It is recommended that any user database to be recovered be dropped and created with the **for load** option

Earlier, the database creation and alteration scripts were saved. These now need to be combined into one script which will re-create the user database with the **for load** option. The **for load** option will insure the success of the **LOAD DATABASE** and **LOAD TRANSACTION** commands.

BulkCopy Utility

The **bcp** (BulkCopy) utility is located in the **$SYBASE**/bin directory and is designed to copy data to and from SQL Server databases to operating system files.

In general, you must supply the following information for transferring data to and from SQL Server:

- Name of the database and table
- Name of the operating system file
- Direction of the transfer (in or out)

In order to use **bcp**, you must have a SQL Server account and the appropriate permissions on the database tables and operating system files that you will use. To copy data into a table, you must have **insert** permission on that table. To copy data out to an operating system file, you must have select permission on the following tables:

- The table being copied
- sysobjects
- syscolumns
- sysindexes

There are two template scripts located in **$SYBASE/scripts/bcp_scripts**:

- **bcp_out_testbed** is the template script to copy data from a database
- **bcp_inp_testbed** is the template script to copy data into a database.

User Database Recovery Scenario

The database **UserDB** was created using the following script excerpt: (stored in $SYBASE/scripts/create.databases/userdb.sql)

```
create database UserDB
on data_dev1 = 100
log on tx_log1 = 50
```

It was later modified using the following script excerpt: ($SYBASE/scripts/create.databases/alteruserdb.sql)

```
alter database UserDB
on data_dev1 = 50
```

For the purposes of this example, the full database backup and transaction log dumps were successful and located in $SYBASE/backups/backups_for_970101/UserDB.dat and UserDB_tx.dat.

1 At the UNIX prompt, type **cd /usr/ecs/Rel_A/COTS/sybase/scripts/create.databases**, then press **Return.**

2 At the UNIX prompt, type **cp template.sql userdb_for_load.sql**

3 Using the text editor of your choice, open the file created above and insert the following:

create database UserDB on data_dev2=100 log on tx_log2=50 for load
go
alter database UserDB on data_dev3=50
go

4 When complete, save the file.

5 At the UNIX prompt, type:

isql -Usa -S*servername* -iuserdb_for_load.sql -ouserdb_for_load.out

then press **Return**.

6 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

- Remember the ***SybaseSAPassword*** is case sensitive.
- The database is recreated

7 At the UNIX prompt, type **more userdb_for_load.out**

- This allows you to view the output file for errors.

8 At the UNIX prompt, type **isql -Usa**, then press **Return**.

9 At the ISQL prompt, type **LOAD DATABASE UserDB from "$SYBASE/backups/backups_for_*date*/UserDB.dat**, then press **Return**; type **go**, then press **Return**.

- This loads the database data back into the database.

10 At the ISQL prompt, type **LOAD TRANSACTION UserDB from $SYBASE/backups/backups_for_970101/UserDB_tx.dat**, then press **Return**; type **go**, then press **Return**.

- This loads the transaction log back into the database.

This page intentionally left blank.

SQL Server Login Accounts and Privileges

In order to connect to a SQL Server a login must be created by the DBA. That login must then be assigned to particular database(s) that the user will access. All login details are stored in the syslogins table in the **master** database.

Providing access to SQL servers and their databases consists of the following steps:

- A server login account for a new user is created.
- The user is added to a database and optionally assigned to a group.
- The user or group is granted permissions on specific commands and database objects.

The procedure involves creating a script that uses three SQL stored procedures:

- **sp_addlogin -** adds a new login name to the master database but does not grant access to any user database.
- **sp_adduser -** adds access capability to the defined database(s).
- **sp_helpuser -** checks the master database for information about database user logins on the system.

By creating and storing the script file, you can easily restore all database users and their assigned databases in case of catastrophic system failure.

Receive Approval For Account Creation

The SQL Server Logins and Privileges process (Figure 7) begins when the requester fills out a "SQL Server Login Account Request Form" and submits it to his or her supervisor (Figure 8).

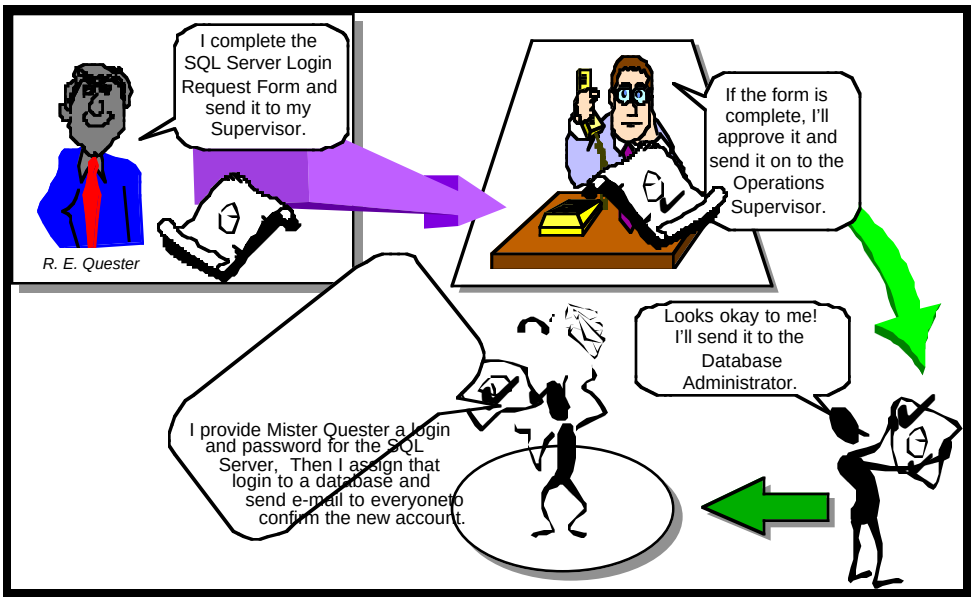

Figure 7. SQL Server Login Approval Process

SQL Server Login Account Request

REQUESTER INFORMATION:

Name: _____

UNIX ID: _____ Group: _____

Office Phone Number: _____

E-Mail Address: _____ Office Location: _____

Database(s) to be accessed: _____

_____

_____

Permissions require for database objects:_____

_____

Justification: _____

_____

_____

Date of Request: _____ Date Required: _____

Supervisor Approval: _____ Date: _____

Ops Supervisor Approval: _____ Date: _____

Figure 8. SQL Server Login Account Request Form

If it is complete and if the request for a new or modified SQL Server login account fits within policy guidelines, the Ops Super approves the request and forwards the request form to the DBA to implement. After the user's login account is created, the DBA provides the user with a password to use for logging onto their SQL Server login. The user can change the initial

password if local DAAC policy requires, or if he/she prefers to select his/her own password. The DBA also sends an e-mail message to the user's supervisor informing him/her that the user's SQL Server login was created.

Create SQL Server Login Accounts

The second step in creating the fully functional database user involves the DBA assigning a SQL Server logname and password for the user. The **sp_addlogin** command is used to perform this task. It will be included in the creation script as part of the procedure.

Add User to Database(s)

After the DBA determines the logname and password for the new user, the user must be added to the databases that he/she will use. The **use** and **sp_adduser** commands are used to perform this task. They will be included in the creation script as part of this procedure.

Create an SQL Server Login Procedure

1 Log into the server on which the new user database is to be created by typing: **telnet** ***ServerName*** or **rlogin** ***ServerName*** or **rsh** ***ServerName***, then press **Return**.

- If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
- If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY** ***IPNumber*****:0.0** or **setenv DISPLAY** ***ServerName*****:0.0**, then press **Return**.

5 At the UNIX prompt, type **su sybase**, then press **Return**.

- A password prompt is displayed.

6 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

- Your new home directory is **/usr/ecs/Rel_A/COTS/sybase** and all required environment variables have been set.

7 At the UNIX prompt, type **cd scripts/create.users**, then press **Return**.

8 At the UNIX prompt, type **cp template.sql *UserName*.sql**, then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the new user login.

9 Using the text editor of your choice, edit ***UserName*.sql** (Figure 9) and make changes to information enclosed in brackets (including the brackets) as appropriate:

```
/*****************************************/
/* name: [template.sql]                  */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/
sp_addlogin [login], [password]
go
use [default database]
go
sp_adduser [login]
go
sp_helpuser
go
```

Figure 9. Sample template.sql File for New Database User Login

10 After the changes have been made, save the file according to the rules of the text editor.

11 At the UNIX prompt, type:

isql -Usa -S*ServerName* -i*UserName*.sql -o*UserName*.out

then press **Return**.

- ***ServerName*** is the name of the server on which the **master** database is stored.
- ***UserName.sql*** is the name of the script file you created in step 9.
- ***UserName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

12 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

- Remember the ***SybaseSAPassword*** is case sensitive.

13 When the UNIX prompt is again displayed the process is complete.

14 At the UNIX prompt, type **more *UserName.out*** , then press **Return**.

- This allows you to view the ***UserName.out*** file to confirm that the user login has been reated or to check for user login creation errors.

Granting or Revoking Database Access Privileges

Permissions are used to control access within a database. The DBA uses the **grant** and **revoke** statements to accomplish this. There are two types of permissions within a database, **Object** and **Command**. In general, **Object** privileges control select, insert, update, delete, and execute permissions on tables, views, and stored procedures and **Command** permissions (which really do not apply to the Testbed) control access to the **CREATE** statements for databases, defaults, procedures, rules, tables, and views.

The syntax for the **grant** and **revoke** statements are quite similar:

grant *privilege(s)* **to** [public | name_list | role_name]

revoke *privilege(s)* **from** [public | name_list | role_name]

The privileges that may be granted or revoked include:

- select
- update
- insert
- delete
- references
- executeIt is recommended that the DBA store the privilege granting and revoking commands in ".sql" files in the **$SYBASE/scripts/create.db_objects** directory along with their results.

Grant or Revoke Database Access Privileges Procedure

1 Determine the privileges that you want to grant and to which user(s).

2 Log into the server on which the new user database is to be created by typing: **telnet *ServerName*** or **rlogin *ServerName*** or **rsh *ServerName***, then press **Return**.

- If you use **telnet**, a **Login:** prompt appears. Continue with Step 2 below.
- If you use **rlogin** or **rsh**, the system uses the same logname currently in use. Skip to Step 4 below.

3 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

4 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

5 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

6 At the UNIX prompt, type **su sybase**, then press **Return**.

- A password prompt is displayed.

7 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/Rel_A/COTS/sybase** and all required environment variables have been set.

8 At the UNIX prompt, type **isql -Usa**, then press **Return**.

9 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

- Remember the ***SybaseSAPassword*** is case sensitive.

10 If you are granting privileges, at the ISQL prompt, type **grant *Privilege* to *Logname***, then press **Return**.

- If more than one privilege is to be granted, repeat this step on subsequent lines.

11 If you are revoking privileges, at the ISQL prompt, type **revoke *Privilege* from *Logname***, then press **Return**.

- If more than one privilege is to be revoked, repeat this step on subsequent lines.

12 When all privileges have been assigned, type **go**, then press **Return**.

Database Tuning and Performance Monitoring

Day-to-day operation of a complex database system requires the DBA to actively and continuously monitor the performance of the database servers for degradation of processing speed, available disk space, and connectivity, to name just a few of the many configurable parameters. This section discusses some of the SQL tools that are available to assist the DBA in performing system tuning and performance monitoring.

Configure SQL Server

System 10 SQL Server has about 40 configurable parameters that can be set permanently or for single runs of a program. Using the **sp_configure** command, you can display a list of the names and current values of the parameters (Figure 10). Please refer to the Sybase System Administrator's guide (URL = `http://sybooks.sybase.com/cgi-bin/nph-dynaweb/srv11007/sysadm/1.toc`) for specific information about what each of the parameters controls.

The column titled **name** is the description of the variable. The column titled **minimum** is the minimum allowable configuration setting. The column titled **maximum** is the theoretical maximum value to which the configuration option can be set. The actual maximum value is dependent on the specific platform and available resources to the SQL Server. The column titled **config_value** reflects the current system default values. The column titled **run_value** reflects the values the system is currently utilizing, which may be different from the default values.

```
name                          minimum     maximum          config_value  run_value
----------------------------- ----------- ---------- ------------ -----------
recovery interval                       1       32767            0           5
allow updates                           0           1            0                   0
user connections                        5  2147483647            0          25
memory                               3850  2147483647            0        5120
open databases                          5  2147483647            0          12
locks                                5000  2147483647            0        5000
open objects                          100  2147483647            0         500
procedure cache                         1          99            0          20
fill factor                             0         100            0                   0
time slice                             50        1000            0         100
database size                           2       10000            0           2
tape retention                          0         365            0                   0
recovery flags                          0           1            0                   0
nested triggers                         0           1                1              1
devices                                 4         256            0          10
remote access                           0           1                1              1
remote  logins                          0  2147483647            0          20
remote sites                            0  2147483647            0          10
remote connections                      0  2147483647            0          20
pre-read packets                        0  2147483647            0           3
upgrade version                         0  2147483647         1002                1002
default  sortorder id                   0         255           50                  50
default language                        0  2147483647            0                   0
language in cache                       3         100            3                   3
max online engines                      1          32            1                   1
min online engines                      1          32            1                   1
engine adjust interval                  1          32            0                   0
cpu flush                               1  2147483647          200                 200
i/o flush                               1  2147483647         1000                1000
default character set id                0         255            1                   1
stack size                          20480  2147483647            0          28672
password expiration interval            0       32767            0                   0
audit queue size                        1       65535          100                 100
additional  netmem                      0  2147483647            0                   0
default network packet size           512      524288            0         512
maximum network packet size           512      524288            0         512
extent i/o buffers                      0  2147483647            0                   0
identity burning set factor             1     9999999         5000                5000
allow  sendmsg                          0           1            0                   0
sendmsg starting port number            0       65535            0                   0
```

Figure 10. **sp_configure** ***Sample Output***

In order to perform the procedure, the DBA must have obtained the following information:

- Name of server to be configured
- New values for configuration variables.

To set SQL Server configuration variables, the DBA must have **sa_role** (SQL Server) permissions. To set SQL Server configuration variables **allow updates**, **audit queue size**, **password expiration interval**, or **remote access**, **sso_role** (SQL Server) is also required.

Some parameter values take affect as soon as you reset the value. Others, which involve memory reallocation, do not change until you reset the value and then reboot SQL Server.

Configure SQL Server Procedure

1 Log into the server which will be reconfigured by typing: **telnet *ServerName*** or **rlogin *ServerName***, then press **Return**.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- Remember that ***YourPassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

5 Begin the SQL session by typing at the UNIX prompt **isql -S*ServerName***, then press **Return**.

6 Type **sp_configure**, press **Return**, type **go**, then press **Return.**

- A list of the configurable parameters, their maximum and minimum values, the current setting and the default values is displayed.

7 After determining which parameter(s) to reset, type:

sp_configure *"ParameterName", NewValue*

then press **Return.**

- ***ParameterName*** is the name from the list displayed in step 6 above. Be sure to enclose the name in double quotes.
- ***NewValue*** is the numeric value that you want to assign to ***ParameterName***.

8 Repeat step 7 above for each parameter you want to reconfigure.

9 When complete, type **quit** at the ISQL prompt, then press **Return**.

- You are returned to the UNIX prompt.

Memory Utilization and Configuration

Monitoring and configuring memory is probably the most important of the configurable parameters. If more data can be accessed and worked on in the memory cache, then the system will perform faster. However, configuring a large amount of the cache to an underutilized procedure can lead to a degradation in system performance.

When SQL server starts up, it refers to the **memory** system configuration (Figure 10) to determine the number of pages of real memory that the SQL server requests from the operating system when it comes up.

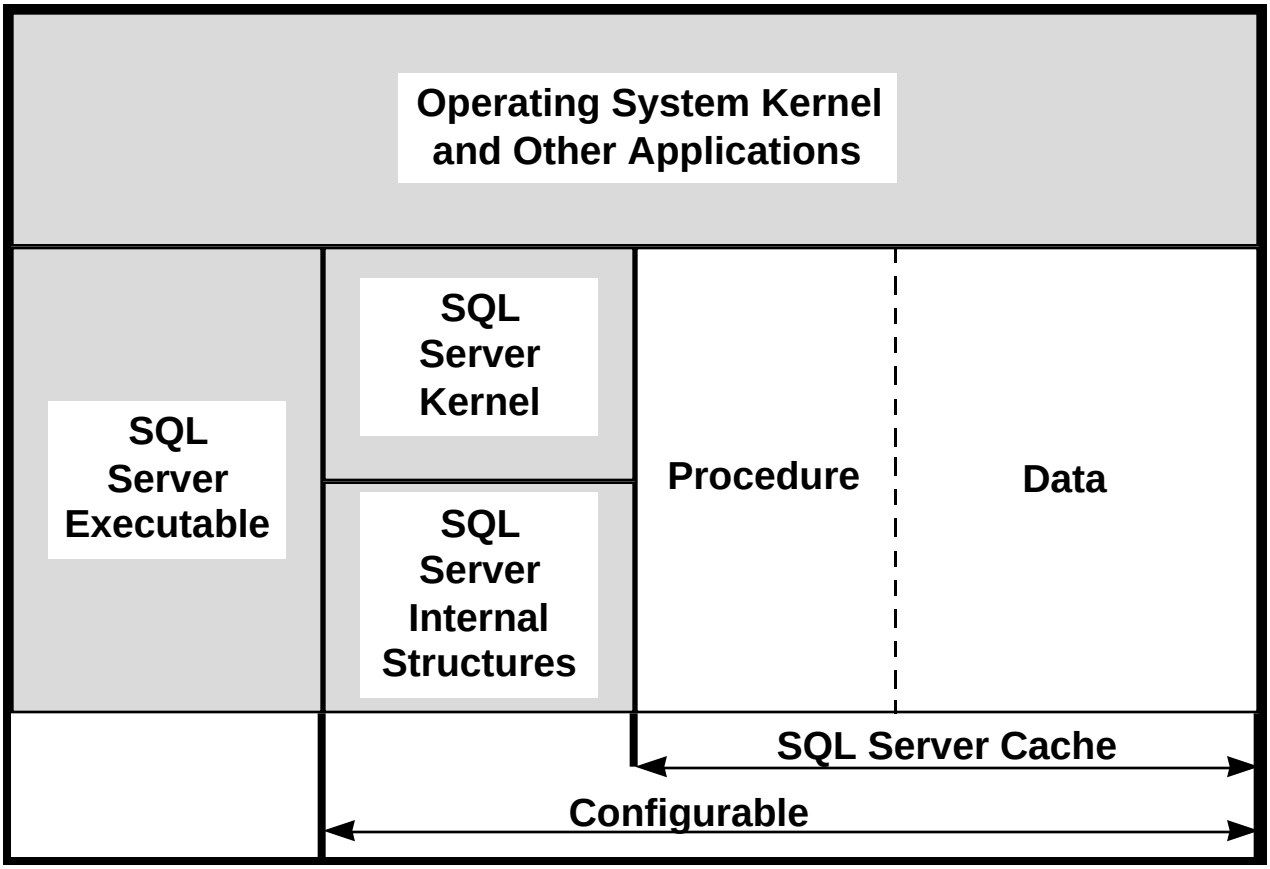

Figure 11. Physical Memory Allocation

An SQL Server is configured when the Database Administrator determines that a customization or fine tuning is required to optimize memory allocation or performance.

Integrity Monitoring

The Database Consistency Checker (**dbcc**) is a set of utility commands for checking the logical and physical consistency of a database. Use the **dbcc** commands:

- As part of regular database maintenance (periodic checks run by the System Administrator). These checks can detect, and often correct, errors before they affect a user's ability to use SQL Server.
- To determine the extent of possible damage after a system error has occurred.
- Before backing up a database.
- Because you suspect that a database is damaged. For example, if using a particular table generates the message "Table corrupt", you can use dbcc to determine if other tables in the database are also damaged.

The integrity of the internal structures of a database depends upon the System Administrator or Database Owner running database consistency checks on a regular basis. Two major functions of dbcc are:

- Checking allocation structures (the commands checkalloc, tablealloc, and indexalloc.
- Checking page linkage and data pointers at both the page level and row level (checktable and checkdb). The next section explains page and object allocation and page linkage.

dbcc is used in the database backup scripts to determine if the database is properly configured and without errors. If errors occur, the backup will not proceed and a message is sent to the DBA informing him/her of the problem.

This page intentionally left blank.

Troubleshooting

Diagnosing Database System Problems

Symptoms of a Damaged master Database

A damaged master database can be caused by a media failure in the area on which master is stored, or by internal corruption in the database. A damaged master database makes itself known in one or more of these ways:

- SQL Server cannot start.
- There are frequent or debilitating segmentation faults or input/output errors.
- **dbcc** (the database consistency checker) reports damage during a regularly-scheduled check of your databases.

This page intentionally left blank.

Practical Exercises

Introduction

These practical exercises are presented in "day-in-the-life" scenarios relating to database administration activities. They represent real situations that you, as database administrator, are likely to encounter on a day-to-day basis.

Equipment and Materials

A functioning Pre-Release B Testbed system.

Starting and Stopping SQL Server

1. Check to see that the SQL Server processes are running. If they are not running, start them. If they are running, shut them down and restart them.

Database Devices

1 Create the script file(s) that will create a database device on the server to be announced from the podium. The following parameters should be met:

- Device Name = class_device
- Physical Name = TBA
- Device Size = 100 MB (be sure to convert this to blocks!)
- Virtual Device Number = you must determine this number.

2 Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.

3 If time and space permit (to be determined by the instructor), perform the creation of this device.

User Databases

Margaret Janis, a data scientist, has requested that a small database be created so she can track 50 new experiments.

1 Create the script file(s) that will create a user database on the server to be announced from the podium. The following parameters should be met:

- Database Name = mjanisdb
- Database Device Name = class_device
- Database Size = 50 MB
- Transaction Log Size = 5 MB
- Transaction Log Device = default_device

2 Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.

3 If time and space permit (to be determined by the instructor), perform the creation of this database.

Backup and Recovery

1 Perform a manual backup of the autosys database. Send the backup to the autosys_backup server.

2 Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.

3 Assume that the database device you created at the beginning of these exercises has failed. Perform all the steps required to recreate the database device and restore the database to full functionality.

Slide Presentation

Slide Presentation Description

The following slide presentation represents the slides used by the instructor during the conduct of this lesson.

This page intentionally left blank.